

NAME: Zahid Ali

ROLLNO: K_24SW012

Report: Student management System

SUBMITTED TO: ENGR: Asmatullah Zubair soomro



❖ Student management System Report

➤ Project Overview :

- This project is a Student Management System designed and developed as part of my Data Structure and Algorithm (DSA) course third-semester project. The primary goal was to implement an efficient inventory management system specifically for managing student data using linked lists as the core data structure. The project integrates core CRUD functionalities — Create, Read, Update, Delete — using linked list methods such as Add(), Delete(), while providing a user-friendly GUI with Java Swing.
- Unlike traditional database-driven software, this system dynamically manages student data — adding new students, updating existing data, searching records by ID, deleting entries, displaying all student records, and exporting the data as CSV files — all using a manually implemented linked list.
- The inclusion of a Login Dialog secures the system with a simple username-password authentication before entering the main dashboard.

➤ Objectives :

- To efficiently and dynamically manage student data without using a database, by applying linked list data structures.

- To implement full CRUD functionalities with clearly defined linked list-related methods to demonstrate data structure principles.
- To create a graphical user interface (GUI) that enables easy interaction with the system using buttons for each operation: Add, Delete, Update, Display, Search by ID, and Export CSV.
- To secure access through a login interface controlled by username and password.
- To familiarize with Java Swing GUI design and file handling for CSV export.
- To emphasize practical application of DSA concepts in real-world scenarios for academic learning.

▪ **Technologies Used :**

- Programming Language: Java
- IDE: VS Code
- Data Structure: Linked List (custom implementation)
- GUI: Java Swing components
- File Handling: Java IO for exporting CSV

➤ **Features:**

- **Login Authentication:** Requires correct username ('admin') and password ('2645') before accessing the application.
- **Student Data Management:** Add new student records dynamically to the linked list.
- **CRUD Operations:** Create, Read, Update, Delete, and Display operations handled using linked list methods.
- **CSV Export:** Export all student data into a CSV file for sharing, backup, or analysis.
- **Data Validation:** Prevents adding duplicate student IDs.
- **User-friendly UI:** Clear buttons, input fields, and dialog messages for success/error feedback.

▪ **User Interface:**

- **Login Page:** Simple login screen prompting for username and password. Only authorized users can proceed to the main dashboard.
- **Main Dashboard:** Input fields for ID, Name, Grade, and Attendance. Buttons for Add, Delete, Update, Search, Display, Export CSV, and Exit. Displays data in Table.

▪ **Code Implementation:**

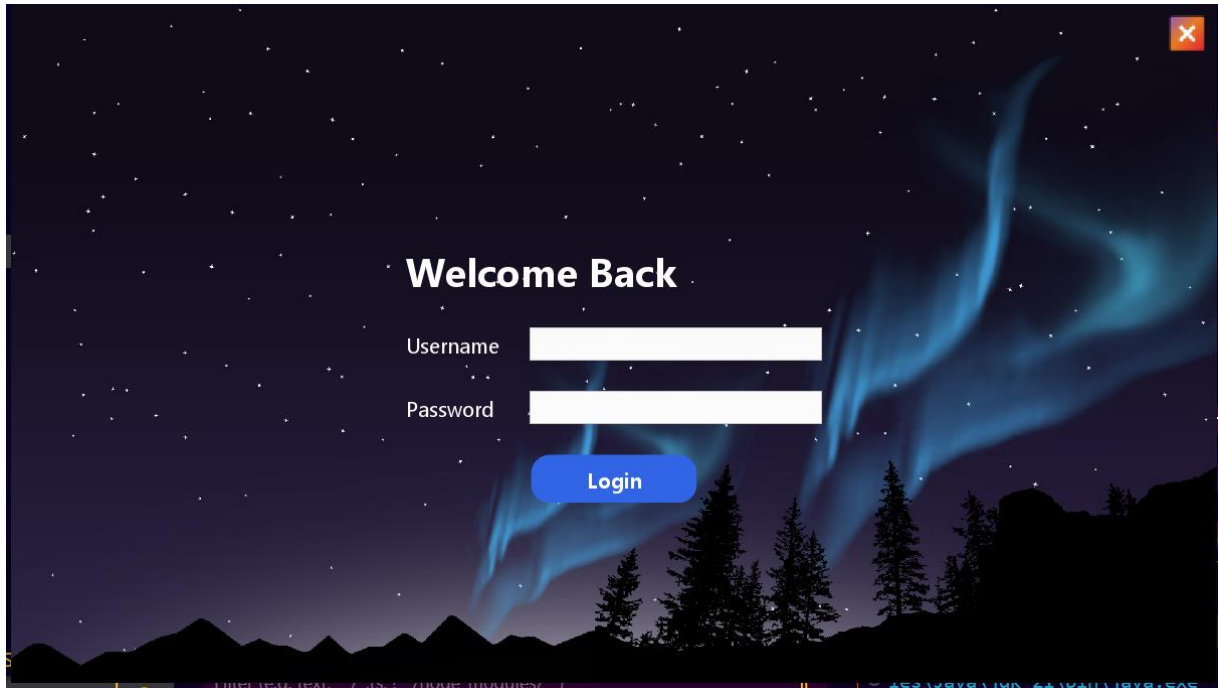
- **LinkedList.java:** Contains singly linked list implementation including Node class and CRUD methods.
- **Main.java:** Manages GUI and connects button events to linked list operations.
- **LoginDialog.java:** Authenticates user login before accessing the main dashboard.
- **Student.java:** Defines Student object with ID, Name, Grade, and Attendance fields.

▪ **Implementation Steps:**

- Set up project in VS Code and configured Java environment.
- Developed linked list structure and implemented CRUD methods.
- Designed GUIs for login and dashboard using Java Swing.
- Linked GUI events to LinkedList methods.
- Added input validation and duplicate checks.
- Tested all operations: login, add, search, delete, update, display, export.
- Finalized UI and dialogs.

❖ Screenshots OF GUI

➤ Screenshot 1: Admin Page:

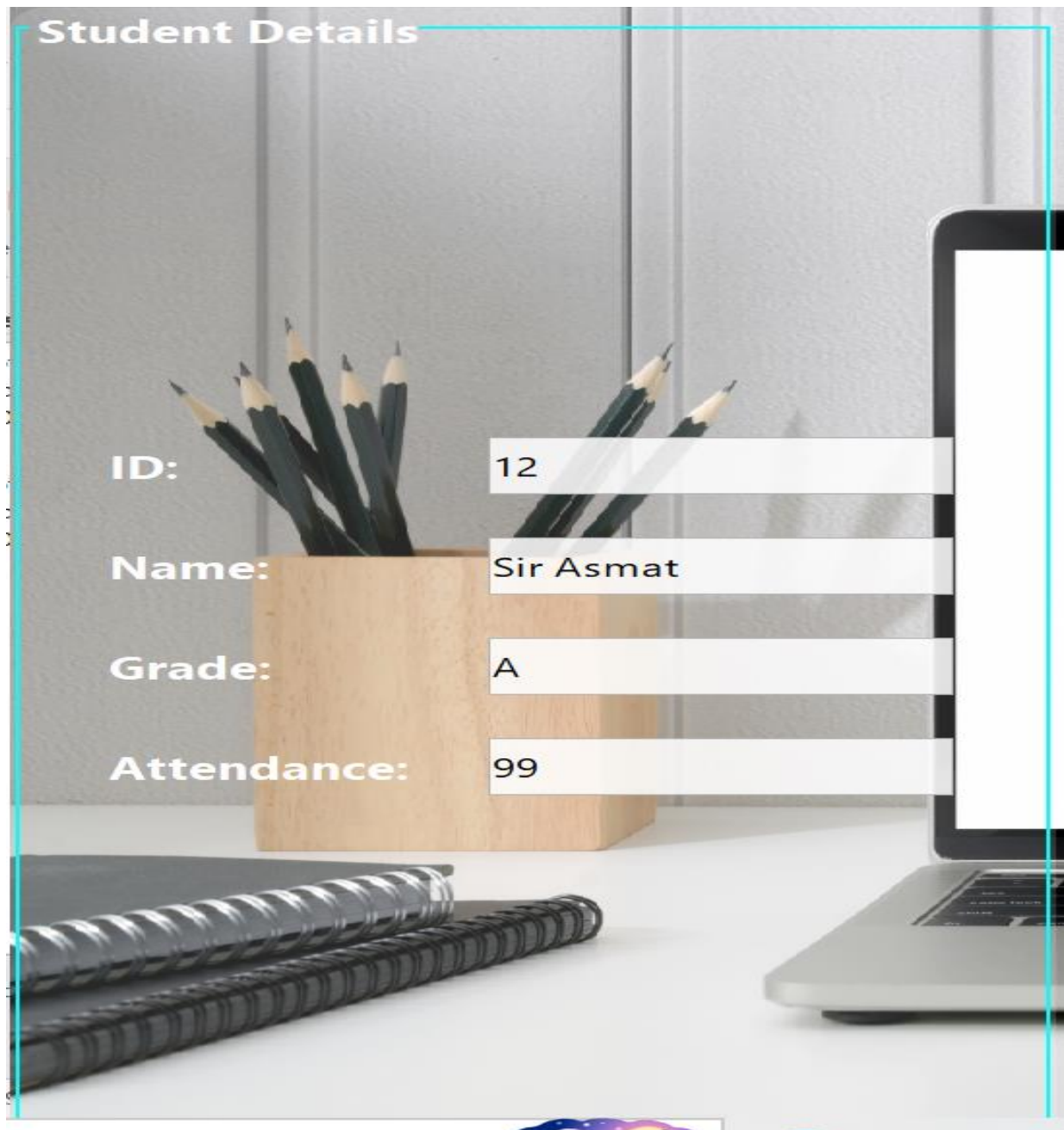


➤ Screenshot 2: Home Page:



➤ **Screenshot 3: Add Record:**

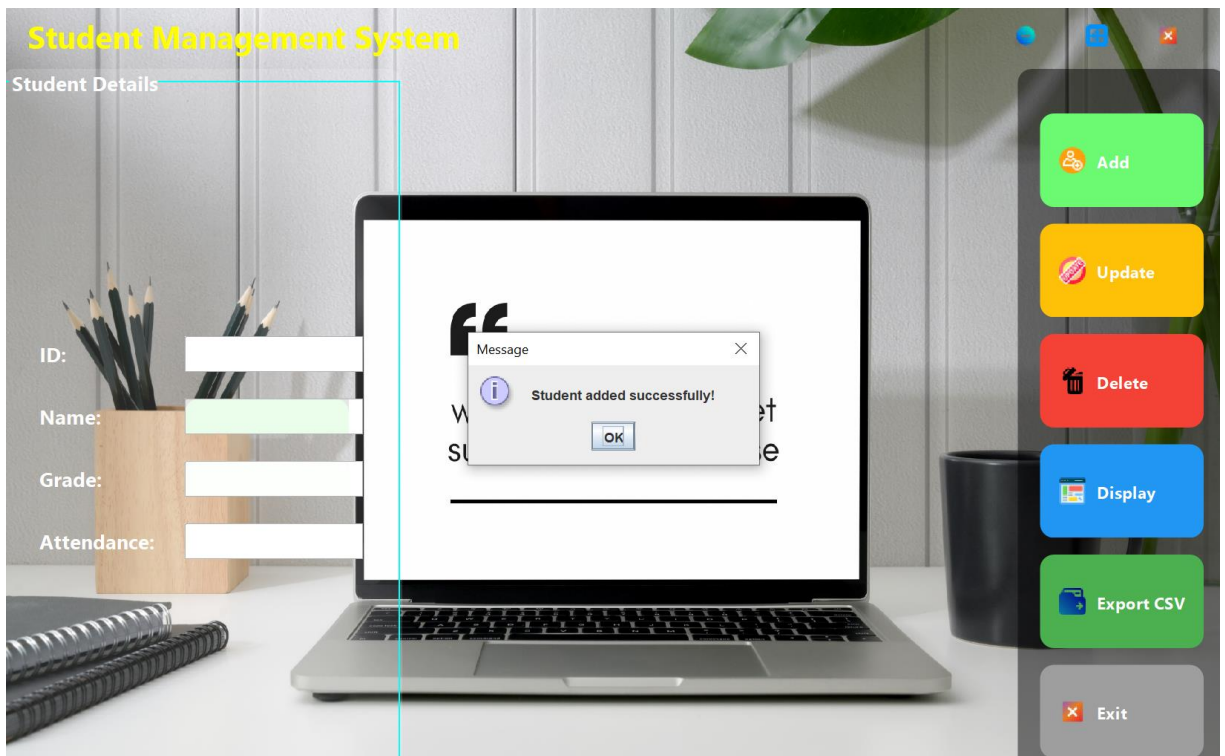
Student Details:



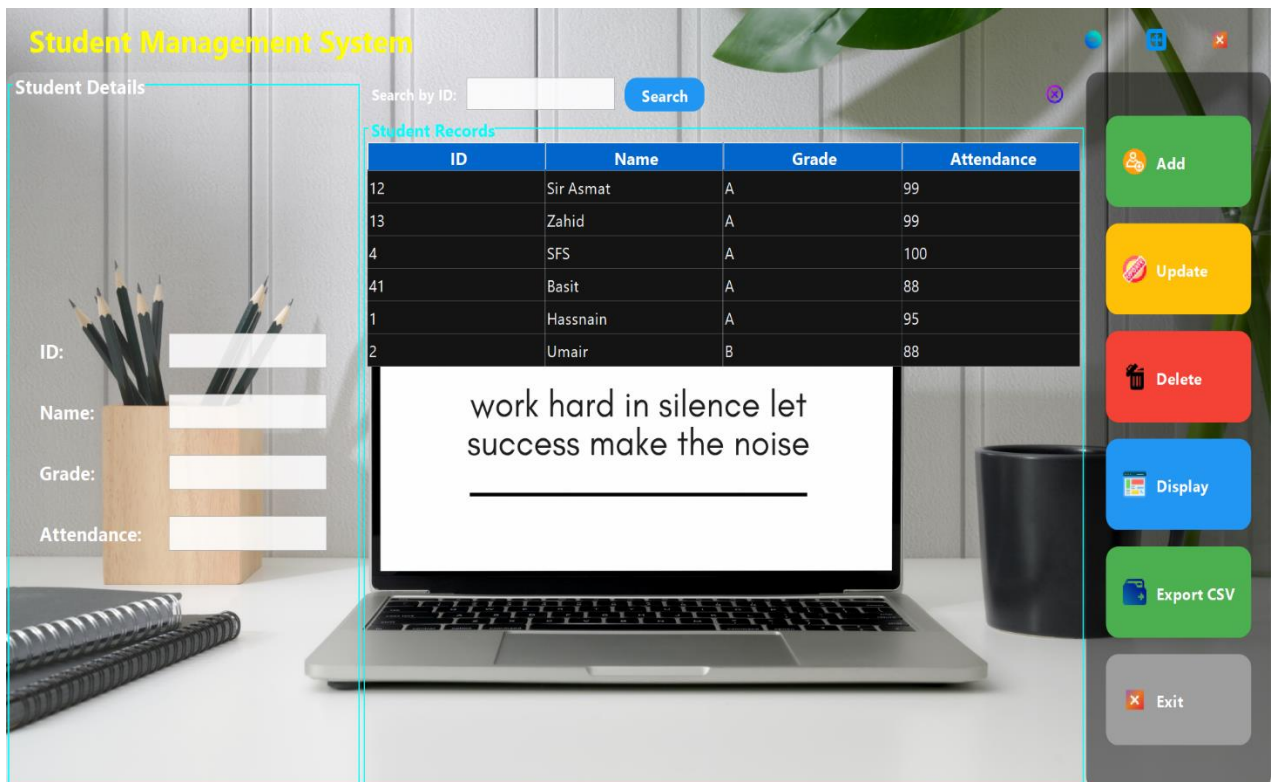
The screenshot shows a 'Student Details' form overlaid on a background image of a desk. The desk features a wooden pencil holder with several black pencils, two spiral-bound notebooks, and a laptop. The form is titled 'Student Details' and contains four input fields with the following data:

Field	Value
ID:	12
Name:	Sir Asmat
Grade:	A
Attendance:	99

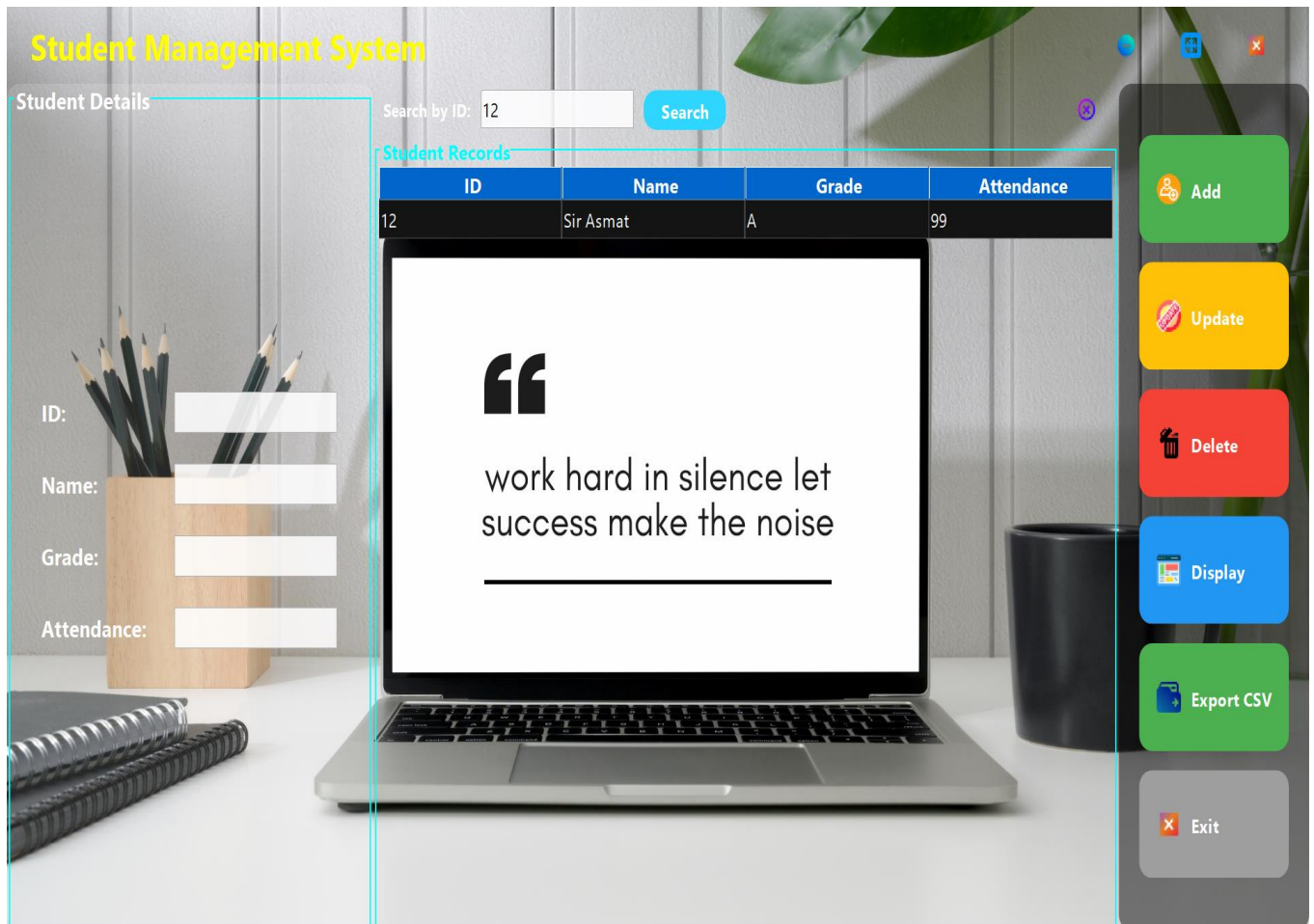
- **Successful Student Record:**



➤ **Screenshot 4: Display**

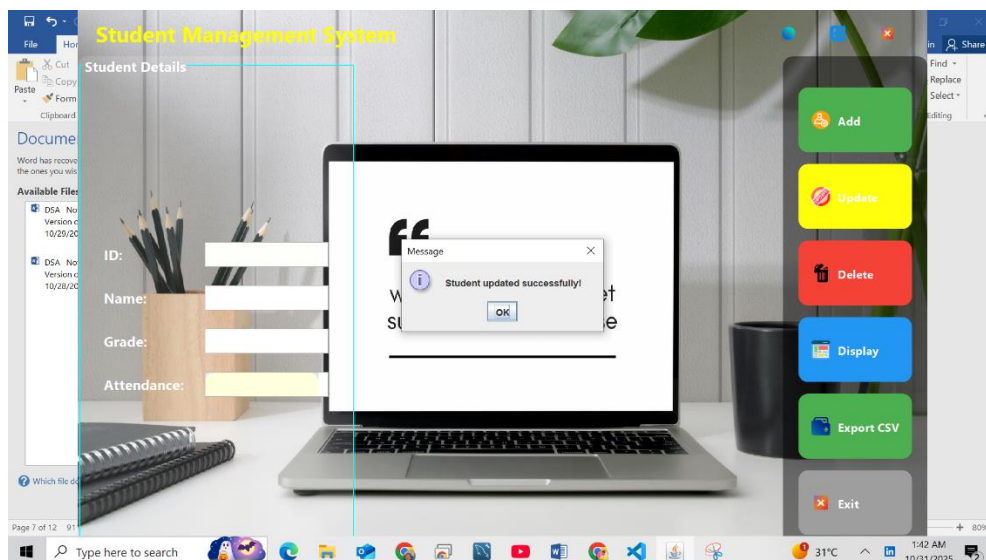


➤ Screenshot 5 : Search By ID

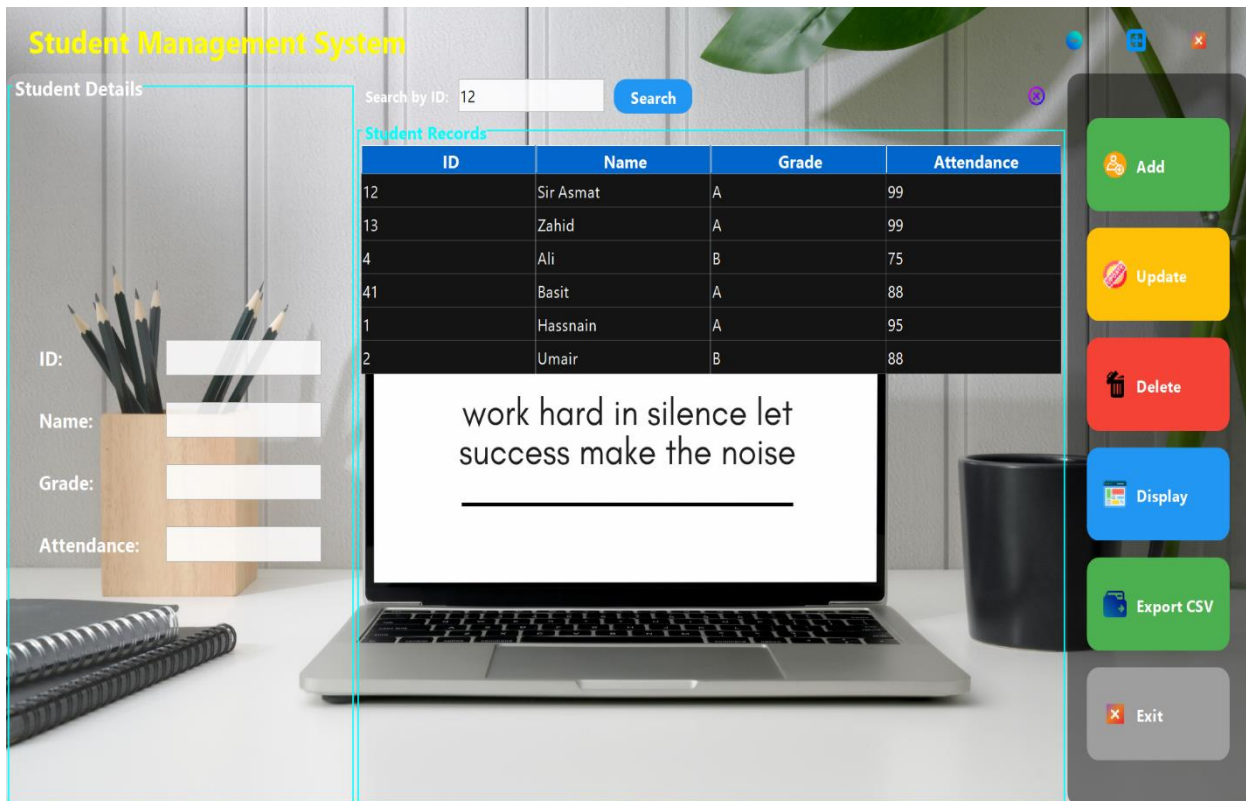


➤ Screenshot 6 : Update

Successful update

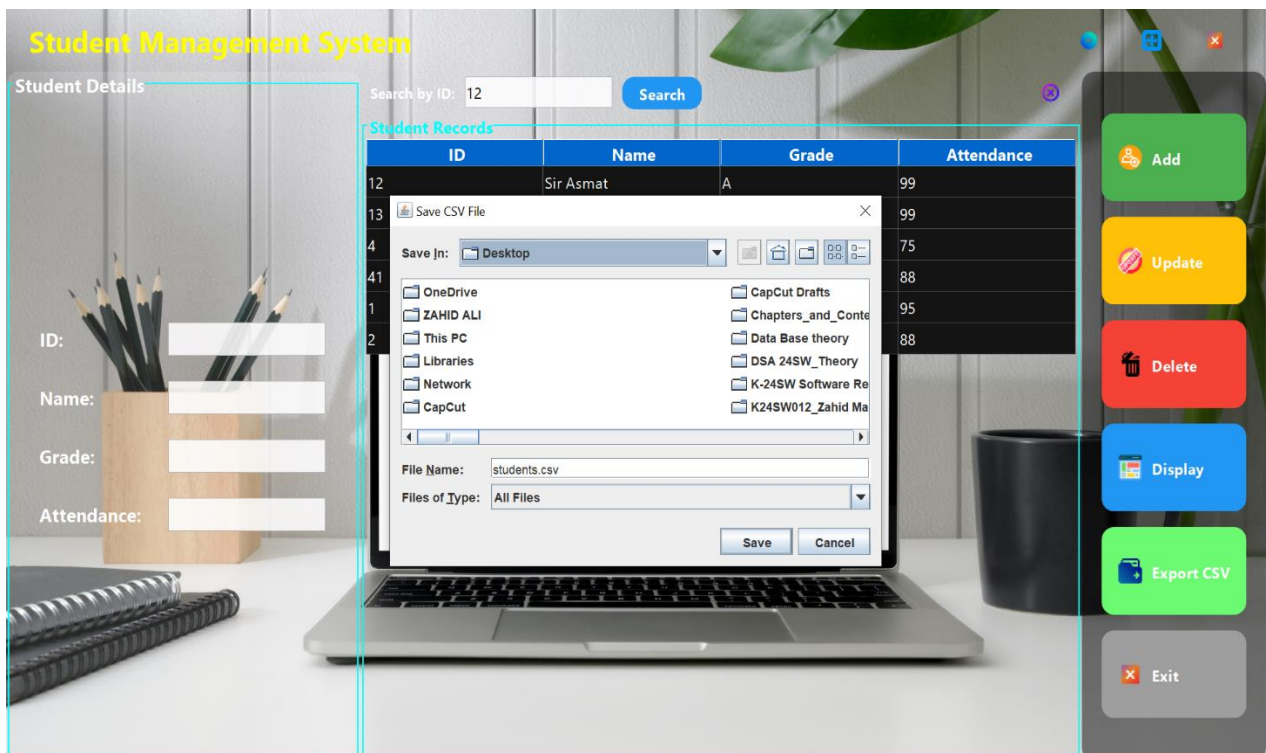


After Update Table Record:



➤ Screenshot 7 : Export CSV

Before Save



Successful Save:

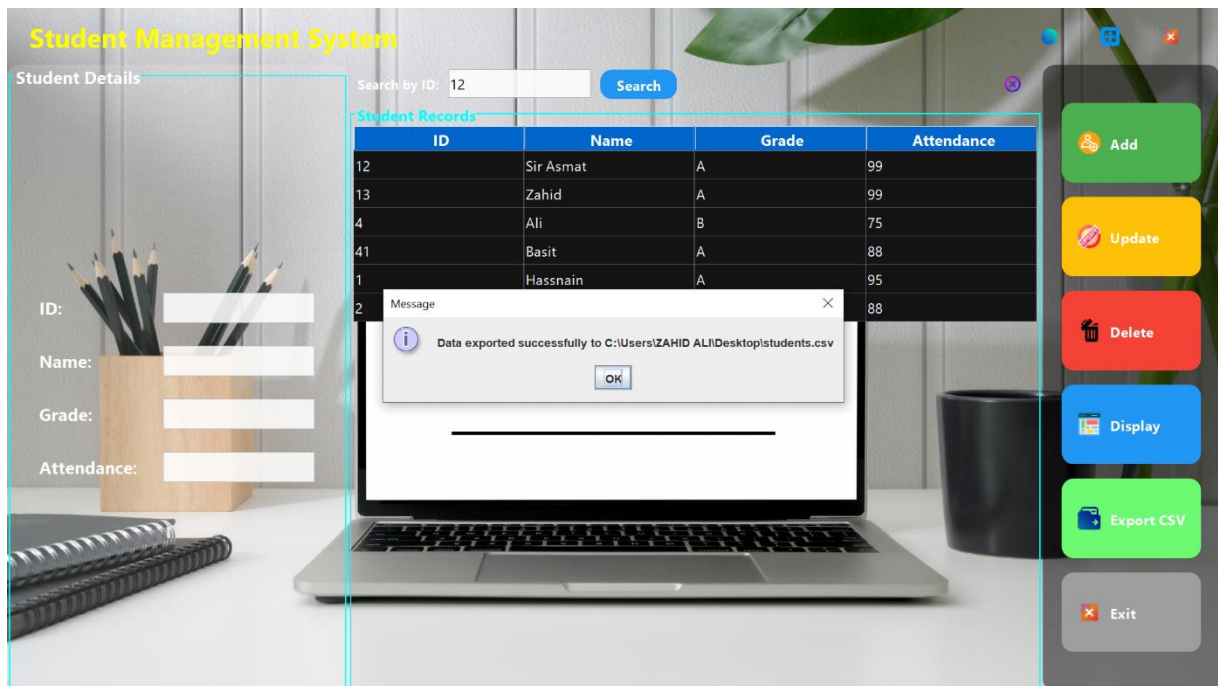


Table data after Save:

The screenshot shows an Excel spreadsheet titled "students - Excel". The data is organized into columns: ID, Name, Grade, and Attendance. The rows contain the following information:

ID	Name	Grade	Attendance
12	Sir Asmat	A	99
13	Zahid	A	99
4	Ali	B	75
41	Basit	A	88
1	Hassnain	A	95
2	Umar	B	88

❖ Code:

▪ LinkedList.java

```
○  
○ public class LinkedList {  
○  
○     // Node inner class  
○     private class Node {  
○         Student student;  
○         Node next;  
○  
○         Node(Student student) {  
○             this.student = student;  
○             this.next = null;  
○         }  
○     }  
○  
○     private Node head;  
○  
○     public LinkedList() {  
○         head = null;  
○     }  
○  
○     // Add Student (at end) with duplicate ID check  
○     public void Add(Student student) {  
○         if (Search(student.getId()) != null) {  
○             throw new IllegalArgumentException("Student with ID " +  
○ student.getId() + " already exists.");  
○         }  
○  
○         Node newNode = new Node(student);  
○  
○         if (head == null) {  
○             head = newNode;  
○         } else {  
○             Node current = head;
```

```

    while (current.next != null) {
        current = current.next;
    }
    current.next = newNode;
}
}

// Search Student by ID
public Student Search(int id) {
    Node current = head;
    while (current != null) {
        if (current.student.getId() == id) {
            return current.student;
        }
        current = current.next;
    }
    return null;
}

// Update Student
public boolean Update(int id, String name, String grade, String
attendance) {
    Node current = head;
    while (current != null) {
        if (current.student.getId() == id) {
            current.student.setName(name);
            current.student.setGrade(grade);
            current.student.setAttendance(attendance);
            return true;
        }
        current = current.next;
    }
    return false;
}
}

```

```
// Delete Student by ID
public boolean Delete(int id) {
    if (head == null) return false;

    if (head.student.getId() == id) {
        head = head.next;
        return true;
    }

    Node current = head;
    while (current.next != null) {
        if (current.next.student.getId() == id) {
            current.next = current.next.next;
            return true;
        }
        current = current.next;
    }

    return false;
}

// Get all students as array
public Student[] GetAll() {
    int count = 0;
    Node current = head;

    while (current != null) {
        count++;
        current = current.next;
    }

    Student[] arr = new Student[count];
    current = head;
    int i = 0;
```



```

○         while (current != null) {
○             arr[i++] = current.student;
○             current = current.next;
○         }
○
○         return arr;
○     }
○
○     // Check empty
○     public boolean isEmpty() {
○         return head == null;
○     }
○ }
○
○
○

```

▪ Main.java

```

▪
▪
▪ import javax.imageio.ImageIO;
▪ import javax.swing.*;
▪ import javax.swing.table.DefaultTableModel;
▪ import javax.swing.table.JTableHeader;
▪ import java.awt.*;
▪ import java.awt.image.BufferedImage;
▪ import java.io.File;
▪ import java.io.FileWriter;
▪ import java.io.IOException;
▪
▪ public class Main extends JFrame {
▪     private LinkedList studentList;
▪     private JTable displayTable;
▪     private DefaultTableModel tableModel;

```

```

private JTextField idField, nameField, gradeField, attendanceField;
private JScrollPane scrollPane;
private JPanel windowButtonPanel;
private JPanel tablePanel; // New panel to hold table and controls

public Main() {
    studentList = new LinkedList();
    initializeGUI();
}

private void initializeGUI() {
    setTitle("Student Management System");
    setSize(1100, 720);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setUndecorated(true);

    // === Background Panel ===
    JPanel backgroundPanel = new JPanel(new BorderLayout()) {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            try {
                Image bg = new
ImageIcon(getClass().getResource("/icon/background.png")).getImage();
                g.drawImage(bg, 0, 0, getWidth(), getHeight(),
this);
            } catch (Exception e) {
                g.setColor(new Color(25, 45, 85));
                g.fillRect(0, 0, getWidth(), getHeight());
            }
        }
    };
    setContentPane(backgroundPanel);

    // === Custom Top Bar ===
    JPanel topBar = new JPanel(new BorderLayout());
    topBar.setOpaque(false);
    topBar.setBorder(BorderFactory.createEmptyBorder(10, 15, 10,
15));

```

```

■      JLabel titleLabel = new JLabel(" Student Management System");
■      titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));
■      titleLabel.setForeground(Color.YELLOW);
■      topBar.add(titleLabel, BorderLayout.WEST);
■
■      // Top right control buttons
■      JPanel controlPanel = new JPanel(new
FlowLayout(FlowLayout.RIGHT, 10, 5));
■      controlPanel.setOpaque(false);
■
■      JButton minBtn = createIconButton("/icon/minimize.png", e ->
setExtendedState(Frame.ICONIFIED));
■      JButton maxBtn = createIconButton("/icon/maximize.png", e -> {
■          if ((getExtendedState() & Frame.MAXIMIZED_BOTH) ==
Frame.MAXIMIZED_BOTH)
■              setExtendedState(Frame.NORMAL);
■          else
■              setExtendedState(Frame.MAXIMIZED_BOTH);
■      });
■      JButton closeBtn = createIconButton("/icon/exit.png", e ->
System.exit(0));
■
■      controlPanel.add(minBtn);
■      controlPanel.add(maxBtn);
■      controlPanel.add(closeBtn);
■
■      topBar.add(controlPanel, BorderLayout.EAST);
■      backgroundPanel.add(topBar, BorderLayout.NORTH);
■
■      // === Left Input Panel ===
■      JPanel inputPanel = new RoundedPanel(25, new Color(255, 255,
255, 40));
■      inputPanel.setLayout(new GridBagLayout());
■      inputPanel.setBorder(BorderFactory.createTitledBorder(
■          BorderFactory.createLineBorder(Color.CYAN, 2),
■          "Student Details", 0, 0, new Font("Segoe UI", Font.BOLD,
18), Color.WHITE));
■      inputPanel.setPreferredSize(new Dimension(360, getHeight()));
■
■      GridBagConstraints gbc = new GridBagConstraints();
■      gbc.insets = new Insets(12, 14, 12, 14);
■      gbc.fill = GridBagConstraints.HORIZONTAL;

```

```

■
■         gbc.gridx = 0; gbc.gridy = 0; inputPanel.add(createLabel("ID:"),
gbc);
■         gbc.gridx = 1; inputPanel.add(idField = createField(), gbc);
■         gbc.gridx = 0; gbc.gridy = 1;
inputPanel.add(createLabel("Name:"), gbc);
■         gbc.gridx = 1; inputPanel.add(nameField = createField(), gbc);
■         gbc.gridx = 0; gbc.gridy = 2;
inputPanel.add(createLabel("Grade:"), gbc);
■         gbc.gridx = 1; inputPanel.add(gradeField = createField(), gbc);
■         gbc.gridx = 0; gbc.gridy = 3;
inputPanel.add(createLabel("Attendance:"), gbc);
■         gbc.gridx = 1; inputPanel.add(attendanceField = createField(),
gbc);
■
■         backgroundPanel.add(inputPanel, BorderLayout.WEST);
■
■         // === Buttons Panel (with icons) ===
■         JPanel buttonPanel = new RoundedPanel(25, new Color(0, 0, 0,
120));
■         buttonPanel.setLayout(new GridLayout(6, 1, 15, 15)); // Updated
to 6 rows for the new export button
■         buttonPanel.setBorder(BorderFactory.createEmptyBorder(40, 20,
40, 20));
■
■         java.util.function.Function<String, ImageIcon> loadIcon = (name)
-> {
■             try {
■                 ImageIcon icon = new
ImageIcon(getClass().getResource("/icon/" + name));
■                 Image scaled = icon.getImage().getScaledInstance(24, 24,
Image.SCALE_SMOOTH);
■                 return new ImageIcon(scaled);
■             } catch (Exception e) {
■                 return null;
■             }
■         };
■
■         JButton addBtn = createModernButton("Add", new Color(76, 175,
80));
■         addBtn.setIcon(loadIcon.apply("add.png"));
■         addBtn.addActionListener(e -> addStudent());
■         buttonPanel.add(addBtn);

```



```

    JButton updateBtn = createModernButton("Update", new Color(255,
193, 7));
    updateBtn.setIcon(loadIcon.apply("update.png"));
    updateBtn.addActionListener(e -> updateStudent());
    buttonPanel.add(updateBtn);

    JButton deleteBtn = createModernButton("Delete", new Color(244,
67, 54));
    deleteBtn.setIcon(loadIcon.apply("delete.png"));
    deleteBtn.addActionListener(e -> deleteStudent());
    buttonPanel.add(deleteBtn);

    JButton displayBtn = createModernButton("Display", new Color(33,
150, 243));
    displayBtn.setIcon(loadIcon.apply("display.png"));
    displayBtn.addActionListener(e -> displayStudents());
    buttonPanel.add(displayBtn);

    JButton exportBtn = createModernButton("Export CSV", new
Color(76, 175, 80)); // New export button
    exportBtn.setIcon(loadIcon.apply("export.png")); // Add an
export icon if available
    exportBtn.addActionListener(e -> exportTableToCSV());
    buttonPanel.add(exportBtn);

    JButton exitBtn = createModernButton("Exit", new Color(158, 158,
158));
    exitBtn.setIcon(loadIcon.apply("exit.png"));
    exitBtn.addActionListener(e -> System.exit(0));
    buttonPanel.add(exitBtn);

    backgroundPanel.add(buttonPanel, BorderLayout.EAST);

    // === Table Panel with Search and Close ===
    tablePanel = new JPanel(new BorderLayout());
    tablePanel.setOpaque(false);
    tablePanel.setVisible(false); // Initially hidden

    // Search Panel
    JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT,
10, 5));

```

```

■ searchPanel.setOpaque(false);
■ JLabel searchLabel = new JLabel("Search by ID:");
■ searchLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));
■ searchLabel.setForeground(Color.WHITE);
■ JTextField searchField = createField();
■ searchField.setPreferredSize(new Dimension(150, 30));
■ JButton searchBtn = createModernButton("Search", new Color(33,
150, 243));
■ searchBtn.addActionListener(e ->
searchStudent(searchField.getText().trim()));
■ searchPanel.add(searchLabel);
■ searchPanel.add(searchField);
■ searchPanel.add(searchBtn);
■
■ // Close Button in top right
■ JButton closeTableBtn = createIconButton("/icon/close.png", e ->
tablePanel.setVisible(false));
■ JPanel topRightPanel = new JPanel(new
FlowLayout(FlowLayout.RIGHT));
■ topRightPanel.setOpaque(false);
■ topRightPanel.add(closeTableBtn);
■
■ JPanel topPanel = new JPanel(new BorderLayout());
■ topPanel.setOpaque(false);
■ topPanel.add(searchPanel, BorderLayout.WEST);
■ topPanel.add(topRightPanel, BorderLayout.EAST);
■
■ tablePanel.add(topPanel, BorderLayout.NORTH);
■
■ // === Table ===
■ String[] columns = {"ID", "Name", "Grade", "Attendance"};
■ tableModel = new DefaultTableModel(columns, 0) {
■     @Override
■     public boolean isCellEditable(int row, int column) {
■         return false; // Make table non-editable
■     }
■ };
■ displayTable = new JTable(tableModel);
■ displayTable.setRowHeight(30);
■ displayTable.setFont(new Font("Segoe UI", Font.PLAIN, 15));
■ displayTable.setForeground(Color.WHITE);

```

```

displayTable.setBackground(new Color(20, 20, 20));
displayTable.setGridColor(new Color(255, 255, 255, 60));
displayTable.setSelectionBackground(new Color(30, 144, 255,
150));
displayTable.setSelectionForeground(Color.WHITE);
displayTable.setShowGrid(true);

JTableHeader header = displayTable.getTableHeader();
header.setFont(new Font("Segoe UI", Font.BOLD, 16));
header.setBackground(new Color(0, 102, 204));
header.setForeground(Color.WHITE);
header.setOpaque(true);

scrollPane = new JScrollPane(displayTable);
scrollPane.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createLineBorder(Color.CYAN, 2),
    "Student Records", 0, 0, new Font("Segoe UI", Font.BOLD,
16), Color.CYAN));
scrollPane.setOpaque(false);
scrollPane.getViewPort().setOpaque(false);

tablePanel.add(scrollPane, BorderLayout.CENTER);
backgroundPanel.add(tablePanel, BorderLayout.CENTER);
setVisible(true);
}

// == Icon Button (for min, max, close)
private JButton createIconButton(String path,
java.awt.event.ActionListener action) {
    JButton btn = new JButton();
    try {
        ImageIcon icon = new
ImageIcon(getClass().getResource(path));
        Image scaled = icon.getImage().getScaledInstance(20, 20,
Image.SCALE_SMOOTH);
        btn.setIcon(new ImageIcon(scaled));
    } catch (Exception e) {
        btn.setText("?");
    }
    btn.setFocusPainted(false);
    btn.setBorderPainted(false);
}

```

```

    btn.setContentAreaFilled(false);
    btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    btn.addActionListener(action);
    return btn;
}

// === UI Helper ===
private JLabel createLabel(String text) {
    JLabel lbl = new JLabel(text);
    lbl.setFont(new Font("Segoe UI", Font.BOLD, 18));
    lbl.setForeground(Color.WHITE);
    return lbl;
}

private JTextField createField() {
    JTextField field = new JTextField();
    field.setFont(new Font("Segoe UI", Font.PLAIN, 15));
    field.setBackground(new Color(255, 255, 255, 220));
    field.setForeground(Color.BLACK);
    field.setCaretColor(Color.BLACK);
    field.setBorder(BorderFactory.createLineBorder(new Color(180,
180, 180), 1, true));
    field.setPreferredSize(new Dimension(160, 32));
    return field;
}

private JButton createModernButton(String text, Color baseColor) {
    JButton btn = new JButton(text) {
        @Override
        protected void paintComponent(Graphics g) {
            Graphics2D g2 = (Graphics2D) g.create();
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
            Color color = getModel().isPressed() ?
baseColor.darker()
                : getModel().isRollover() ? baseColor.brighter()
                : baseColor;
            g2.setColor(color);
            g2.fillRoundRect(0, 0, getWidth(), getHeight(), 25, 25);
            g2.dispose();
            super.paintComponent(g);

```



```

    }
};
btn.setFont(new Font("Segoe UI", Font.BOLD, 15));
btn.setForeground(Color.WHITE);
btn.setFocusPainted(false);
btn.setBorderPainted(false);
btn.setContentAreaFilled(false);
btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
btn.setHorizontalAlignment(SwingConstants.LEFT);
btn.setIconTextGap(10);
return btn;
}

// == Functional Logic (unchanged except for display and search)
private void addStudent() {
    try {
        int id = Integer.parseInt(idField.getText().trim());
        String name = nameField.getText().trim();
        String grade = gradeField.getText().trim();
        String attendance = attendanceField.getText().trim();
        studentList.Add(new Student(id, name, grade, attendance));
        clearFields();
        JOptionPane.showMessageDialog(this, "Student added
successfully!");
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Invalid ID. Please
enter numeric ID.");
    } catch (IllegalArgumentException e) {
        JOptionPane.showMessageDialog(this, e.getMessage());
    }
}

private void updateStudent() {
    try {
        int id = Integer.parseInt(idField.getText().trim());
        String name = nameField.getText().trim();
        String grade = gradeField.getText().trim();
        String attendance = attendanceField.getText().trim();
        if (studentList.Update(id, name, grade, attendance)) {
            clearFields();

```

```

        JOptionPane.showMessageDialog(this, "Student updated
successfully!");
    } else {
        JOptionPane.showMessageDialog(this, "Student not
found.");
    }
} catch (NumberFormatException e) {
    JOptionPane.showMessageDialog(this, "Invalid ID format.");
}
}

private void deleteStudent() {
    try {
        int id = Integer.parseInt(idField.getText().trim());
        if (studentList.Delete(id)) {
            clearFields();
            JOptionPane.showMessageDialog(this, "Student deleted
successfully!");
        } else {
            JOptionPane.showMessageDialog(this, "Student not
found.");
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Invalid ID format.");
    }
}

private void displayStudents() {
    tableModel.setRowCount(0);
    for (Student s : studentList.GetAll()) {
        tableModel.addRow(new Object[]{s.getId(), s.getName(),
s.getGrade(), s.getAttendance()});
    }
    tablePanel.setVisible(true);
    revalidate();
}

private void searchStudent(String idText) {
    try {
        int id = Integer.parseInt(idText);
        Student s = studentList.Search(id);
    }
}

```

```

        if (s != null) {
            tableModel.setRowCount(0);
            tableModel.addRow(new Object[]{s.getId(), s.getName(),
s.getGrade(), s.getAttendance()});
        } else {
            JOptionPane.showMessageDialog(this, "Student not
found.");
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Invalid ID format.");
    }
}

// === New Method: Export Table to CSV ===
private void exportTableToCSV() {
    if (tableModel.getRowCount() == 0) {
        JOptionPane.showMessageDialog(this, "No data to export.
Display students first.");
        return;
    }

    // Prompt user to choose save location
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save CSV File");
    fileChooser.setSelectedFile(new File("students.csv")); //
Default filename
    int userSelection = fileChooser.showSaveDialog(this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        try (FileWriter writer = new FileWriter(fileToSave)) {
            // Write CSV headers
            writer.write("ID,Name,Grade,Attendance\n");

            // Write each row from the table
            for (int i = 0; i < tableModel.getRowCount(); i++) {
                writer.write(tableModel.getValueAt(i, 0) + "," + //
ID
                                tableModel.getValueAt(i, 1) + "," + //
Name
                                tableModel.getValueAt(i, 2) + "," + //
Grade

```

```

        tableModel.getValueAt(i, 3) + "\n"); //
Attendance
    }

    JOptionPane.showMessageDialog(this, "Data exported
successfully to " + fileToSave.getAbsolutePath());
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "Error exporting
data: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void clearFields() {
    idField.setText("");
    nameField.setText("");
    gradeField.setText("");
    attendanceField.setText("");
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new LoginDialog(null, () -> new
Main()));
}
}

// === Rounded Panel ===
class RoundedPanel extends JPanel {
    private final int cornerRadius;
    private final Color backgroundColor;

    public RoundedPanel(int cornerRadius, Color backgroundColor) {
        this.cornerRadius = cornerRadius;
        this.backgroundColor = backgroundColor;
        setOpaque(false);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

```



```

■         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
■         g2.setColor(backgroundColor);
■         g2.fillRoundRect(0, 0, getWidth(), getHeight(), cornerRadius,
cornerRadius);
■     }
■ }
■
■
■

```

■ LoginDialog.java

```

■
■ import javax.swing.*;
■ import java.awt.*;
■ import java.security.AccessController;
■ import java.security.PrivilegedAction;
■
■ public class LoginDialog extends JDialog {
■     private JTextField usernameField;
■     private JPasswordField passwordField;
■     private Runnable onSuccess;
■
■     public LoginDialog(Frame parent, Runnable onSuccess) {
■         super(parent, "Login", false);
■         this.onSuccess = onSuccess;
■
■         setUndecorated(true);
■         setSize(800, 450);
■         setLocationRelativeTo(parent);
■         setBackground(new Color(0, 0, 0, 0));
■
■         // ↻ Background Panel
■         JPanel backgroundPanel = new JPanel(new BorderLayout()) {
■             @Override
■             protected void paintComponent(Graphics g) {
■                 super.paintComponent(g);
■                 try {

```

```

        Image bg = new
        ImageIcon(getClass().getResource("/icon/login_bg.png")).getImage();
        g.drawImage(bg, 0, 0, getWidth(), getHeight(),
        this);
    } catch (Exception e) {
        g.setColor(new Color(35, 55, 110));
        g.fillRect(0, 0, getWidth(), getHeight());
    }
    };
    add(backgroundPanel);

    // ✕ Close Button (resized icon)
    ImageIcon icon = new
    ImageIcon(getClass().getResource("/icon/exit.png"));
    Image scaled = icon.getImage().getScaledInstance(30, 30,
    Image.SCALE_SMOOTH); // resize icon
    JButton closeBtn = new JButton(new ImageIcon(scaled));

    closeBtn.setContentAreaFilled(false);
    closeBtn.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
    closeBtn.setFocusPainted(false);
    closeBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    closeBtn.setOpaque(false);
    closeBtn.addActionListener(e -> dispose());

    // Add to top-right corner
    JPanel topPanel = new JPanel(new BorderLayout());
    topPanel.setOpaque(false);
    topPanel.add(closeBtn, BorderLayout.EAST);
    backgroundPanel.add(topPanel, BorderLayout.NORTH);

    // 🗝 Login Form
    JPanel loginPanel = new JPanel(new GridBagLayout());
    loginPanel.setOpaque(false);
    loginPanel.setBorder(BorderFactory.createEmptyBorder(30, 60, 30,
    60));
    backgroundPanel.add(loginPanel, BorderLayout.CENTER);

    GridBagConstraints gbc = new GridBagConstraints();

```

```
■ gbc.insets = new Insets(10, 10, 10, 10);
■ gbc.fill = GridBagConstraints.HORIZONTAL;
■
■ JLabel titleLabel = new JLabel("Welcome Back");
■ titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 26));
■ titleLabel.setForeground(Color.WHITE);
■ gbc.gridx = 0;
■ gbc.gridy = 0;
■ gbc.gridwidth = 2;
■ loginPanel.add(titleLabel, gbc);
■
■ JLabel userLabel = new JLabel("Username");
■ userLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
■ userLabel.setForeground(Color.WHITE);
■ gbc.gridwidth = 1;
■ gbc.gridy = 1;
■ loginPanel.add(userLabel, gbc);
■
■ usernameField = new JTextField(16);
■ usernameField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
■ usernameField.setBackground(new Color(255, 255, 255, 220));
■ usernameField.setBorder(BorderFactory.createLineBorder(new
Color(200, 200, 200), 1));
■ gbc.gridx = 1;
■ loginPanel.add(usernameField, gbc);
■
■ JLabel passLabel = new JLabel("Password");
■ passLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
■ passLabel.setForeground(Color.WHITE);
■ gbc.gridx = 0;
■ gbc.gridy = 2;
■ loginPanel.add(passLabel, gbc);
■
■ passwordField = new JPasswordField(16);
■ passwordField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
■ passwordField.setBackground(new Color(255, 255, 255, 220));
■ passwordField.setBorder(BorderFactory.createLineBorder(new
Color(200, 200, 200), 1));
■ gbc.gridx = 1;
■ loginPanel.add(passwordField, gbc);
■
```

```

    // ✓ Rounded Small Login Button (only this changed)
    JButton loginBtn = new JButton("Login") {
        @Override
        protected void paintComponent(Graphics g) {
            Graphics2D g2 = (Graphics2D) g.create();
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
            Color color = getModel().isPressed() ? new Color(30, 80,
210)
                : getModel().isRollover() ? new Color(80, 130,
255)
                : new Color(50, 100, 230);
            g2.setColor(color);
            g2.fillRoundRect(0, 0, getWidth(), getHeight(), 25, 25);
            g2.dispose();
            super.paintComponent(g);
        }
    };
    loginBtn.setPreferredSize(new Dimension(110, 32));
    loginBtn.setFont(new Font("Segoe UI", Font.BOLD, 14));
    loginBtn.setForeground(Color.WHITE);
    loginBtn.setFocusPainted(false);
    loginBtn.setBorderPainted(false);
    loginBtn.setContentAreaFilled(false);
    loginBtn.setOpaque(false);
    loginBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR
));
    loginBtn.addActionListener(e -> handleLogin());

    gbc.gridx = 0;
    gbc.gridy = 3;
    gbc.gridwidth = 2;
    gbc.fill = GridBagConstraints.NONE; // stops stretching
    gbc.anchor = GridBagConstraints.CENTER;
    loginPanel.add(loginBtn, gbc);

    setVisible(true);
}

private void handleLogin() {
    AccessController.doPrivileged((PrivilegedAction<Void>) () -> {

```

```

    String u = usernameField.getText().trim();
    String p = new String(passwordField.getPassword());
    if (u.equals("admin") && p.equals("2645")) {
        if (onSuccess != null) onSuccess.run();
        dispose();
    } else {
        JOptionPane.showMessageDialog(this,
            "Invalid username or password!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    return null;
});
}
}

```

▪ Student.java

```

public class Student {
    private int id;
    private String name, grade, attendance;

    public Student(int id, String name, String grade, String
attendance) {
        this.id = id;
        this.name = name;
        this.grade = grade;
        this.attendance = attendance;
    }

    public int getId() { return id; }
    public String getName() { return name; }
    public String getGrade() { return grade; }
    public String getAttendance() { return attendance; }
    public void setName(String n) { this.name = n; }
    public void setGrade(String g) { this.grade = g; }
    public void setAttendance(String a) { this.attendance = a; }

    // toString for display
    @Override

```

```

■      public String toString() {
■          return "ID: " + id + ", Name: " + name + ", Grade: " + grade +
■      ", Attendance: " + attendance;
■      }
■  }
■

```

❖ Conclusion

This Student Management System project effectively demonstrates the power and flexibility of linked lists within a real-world application. It integrates CRUD operations, secure login, and a graphical interface, enhancing both functionality and user experience. The project strengthened my understanding of data structures and their interaction with GUI elements in Java.

❖ Links:

\$ _____ \$

Source Code Repository (GitHub):

<https://github.com/zahidali-dev/Student-Management-System-DSA>

_____ **THE END** _____

\$ _____ **Thank you** _____ \$